UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Métodos de *clustering* para conjuntos de clasificación

Javier Cela López
Tutor: Alberto Suárez González

JULIO 2017

Métodos de *clustering* para conjuntos de clasificación

AUTOR: Javier Cela López
TUTOR: Alberto Suárez González

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2017

Resumen

El objetivo de este Trabajo de Fin de Grado es mejorar la eficacia y la eficiencia de los conjuntos de clasificadores mediante técnicas de *clustering*. .En concreto, se trata de identificar conglomerados o clústeres de clasificadores de acuerdo con sus patrones de predicción en un conjunto de datos independiente del de entrenamiento; por ejemplo, en el conjunto de test o en un conjunto de validación). De entre estos conglomerados podemos seleccionar el que proporciona predicciones más precisas en un conjunto de validación, con la expectativa de que tenga una buena capacidad de generalización. El menor tamaño de este subconjunto de clasificadores implica una mejora de la eficiencia del sistema de predicción. En concreto, se reducen los requisitos de memoria para almacenamiento del sistema y el tiempo de predicción,

Los conjuntos de clasificadores de los que se parte en este estudio son heterogéneos. En concreto, están formados por perceptrones multicapa, árboles aleatorios y en algunas partes del análisis empírico de árboles de decisión de tipo *CART*.

Para la construcción de los conjuntos se utiliza *bagging*. Los problemas de clasificación que se utilizan en la parte de evaluación empírica han sido obtenidos del repositorio web *UCI Machine Learning* (Bache & Lichman, s.f.).

Para realizar el análisis de conglomerados utilizaremos *fuzzy K-Means* y un algoritmo de *clustering* basado en un proceso de templado determinista (*deterministic annealing*) propuesto originalmente por (Bakker & Heskes, 2003). En un primer bloque de experimentos se analiza la homogeneidad de los clústeres identificados por cada uno de estos algoritmos. Para el resto del análisis empírico se utiliza el método propuesto por Bakker & Heskes, ya que, de acuerdo a los resultados obtenidos, permite identificar clústeres homogéneos de manera eficiente.

Finalmente, una vez identificado los clústeres, el estudio se centra en la selección del clúster con mejor capacidad de generalización. La precisión de las predicciones del clúster seleccionado será comparada al conjunto de predictores heterogéneo original y a los subconjuntos homogéneos de los que está compuesto. De acuerdo con los resultados obtenidos, en los problemas de clasificación analizados los clústeres seleccionados obtener tasas de acierto comparables a las de los conjuntos de referencia. Esto indica que, mediante la técnica de *clustering* propuesta, es posible mejorar la eficacia del sistema de predicción, ya que se reducen los requisitos de almacenamiento y se mejoran los tiempos de respuesta

en la etapa de predicción, y, de manera simultánea, se mantienen las tasas de acierto en niveles comparables a los del conjunto original.

No obstante, las mejoras obtenidas conllevan una contrapartida, ya que hay un coste añadido durante el entrenamiento debido a que tanto *bagging* como *clustering* son procesos computacionalmente complejos.

Palabras clave

Aprendizaje automático, análisis de conglomerados, conjuntos de clasificadores, *bagging*, bosques aleatorios.

Abstract

The goal of this work is to improve the efficiency and effectiveness of classifier ensembles using clustering techniques. In particular we aim to identify clusters of classifiers according to pattern of their predictions in a data set that is independent from the one used for training; for instance, in the test set or in a validation set. From these clusters we can select the one that yields the most accurate predictions on a validation set, with the expectation that it will have good generalization capacity. The smaller size of this subensemble entails an improvement of the system's efficiency. In particular, the memory requirements for storage are lowered, and the system's predictions faster.

The original ensembles taken as a starting point in this study are heterogeneous. In particular, they consist of multilayer perceptrons, random trees and, for some parts of the empirical analysis, *CART* decision tress. The ensembles are built using *bagging*. The classification problems analyzed in the empirical evaluation come from the *UCI Machine Learning* repository (Bache & Lichman, s.f.).

As clustering algorithms, we consider *fuzzy K-Means* and an algorithm based on *deterministic annealing*, which was originally proposed by (Bakker & Heskes, 2003). In a first batch of experiments, we analyze the homogeneity of the clusters identified by each of these algorithms. In the remainder of the empirical analysis we use the algorithm introduced by Bakker & Heskes, because, according to the results obtained, it yields clusters that are quite homogeneous in an efficient manner. Once those clusters have been identified, the goal of the study is to select the cluster that has the best generalization capacity. The accuracy of the predictions given by the selected clusters is compared with the original heterogeneous ensemble and with the homogeneous subensembles of which it is composed. According to the results obtained, in the classification problems analyzed, the accuracy rates of the selected clusters are comparable to the reference ensembles. This indicates that using the clustering technique that has been proposed it is possible to improve the efficiency of the classification system: lower storage requirements and faster predictions are obtained. Nonetheless, these improvements do come at a cost: There is some overhead in training, because both *bagging* are *clustering* are computationally costly processes.

Keywords

Machine Learning, clustering analysis, classifier ensembles, bagging, random forests

Agradecimientos

Quisiera aprovechar esta oportunidad para agradecer en la medida de lo posible toda la ayuda y el apoyo que he recibido durante estos años de carrera. Todo el trabajo, el esfuerzo y la dedicación culminan en el presente Trabajo de Fin de Grado.

En primer lugar, por supuesto, agradecer a mi familia todo el apoyo que me siempre me han regalado. A mis padres, por la educación y los valores que me han inculcado. A mis hermanos, Camilo y Cristina, por acompañarme siempre.

A María, por aguantarme durante todos los momentos duros, ofrecerme todo su apoyo incondicional y darme los mejores consejos en cualquier circunstancia.

A mis amigos, porque los momentos de trabajo duro (y los de descanso) se pasan mejor en compañía.

Por último, me gustaría agradecer a mi tutor, Alberto Suárez, toda la dedicación puesta en el desarrollo de mi TFG, ayuda sin la cual no habría sido posible la finalización del mismo.

Javier Cela López

Julio 2017

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1. Introducción

Este trabajo de fin de grado tiene como objetivo investigar en profundidad una propuesta dirigida a conseguir mejorar la eficacia y la eficiencia de conjuntos heterogéneos de clasificadores basados en la identificación de clústeres de clasificadores. Así pues, se enmarca en el área de aprendizaje.

El objetivo del aprendizaje automático es identificar regularidades en conjuntos de datos que permitan bien agrupar dichos datos en grupos que comparten características, como en el análisis de conglomerados (o *clustering*), bien inducir modelos predictivos, como en los problemas de clasificación y regresión. En concreto, la inducción de modelos de predicción a partir de datos es abordada mediante algoritmos de aprendizaje supervisado. Como entrada para este tipo de algoritmos se utiliza un conjunto de N ejemplos etiquetados $\boldsymbol{D} = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \dots, \boldsymbol{e}_N\}$. El conjunto de datos etiquetados puede ser agrupado en una matriz de tamaño $N \times (P+1)$. Cada una de las filas de esta matriz corresponde a un ejemplo del conjunto, representado por el par $\boldsymbol{e}_n = (\mathbf{x}_n, y_n)$, donde $\mathbf{x}_n$ es el vector de atributos e y_n es la etiqueta de clase. Los P atributos que caracterizan cada uno de los ejemplos se agrupan en un vector de P dimensiones $\mathbf{x_n} = (x_{n1}, x_{n2}, \dots, x_{nP})$. Los problemas en los que los que la etiqueta de clase, y_n, toma valores discretos se denominan de clasificación. Aquellos en los que el valor de y_n es real, se denominan de regresión. En este trabajo nos centraremos en problemas de clasificación. El objetivo del aprendizaje automático supervisado es generar, a partir de un conjunto datos de entrenamiento $\boldsymbol{D}_{train}$, un predictor capaz de asignar etiquetas de clase a nuevos ejemplos, que serán en general distintos e independientes de los utilizados para la inducción.

En la práctica, partiremos de un conjunto de datos etiquetado $\boldsymbol{D}$, que posteriormente dividiremos de manera aleatoria en un conjunto de entrenamiento ($\boldsymbol{D}_{train}$) y un conjunto de test ($\boldsymbol{D}_{test}$). El conjunto $\boldsymbol{D}$ se compone de $N+M$ ejemplos (filas) y $P+1$ variables descriptivas (columnas; P atributos y una clase) El conjunto $\boldsymbol{D}_{train}$, que contiene N ejemplos, es utilizado para construir los predictores. El conjunto $\boldsymbol{D}_{test}$, de tamaño M, se utilizará para validación de los modelos construidos. Este proceso de validación puede ser realizado mediante un particionado simple, en el cual se divide $\boldsymbol{D}$ en dos partes de tamaños

prefijados (típicamente 2/3 para $\boldsymbol{D}_{train}$ y 1/3 para $\boldsymbol{D}_{test}$), o mediante un particionado cruzado. En este tipo de validación el conjunto $\boldsymbol{D}$ es dividido en k subconjuntos distintos. De manera iterativa, de utilizan los datos en uno de estos k conjuntos para estimar la tasa de acierto del predictor con el resto de los datos. Estas técnicas se conocen como validación simple y validación cruzada, respectivamente.

El entrenamiento, por su parte, se realiza de manera distinta según el tipo de predictor a construir. Denotaremos un predictor mediante h de tal forma que sea una función del espacio de atributos hacia el espacio de clases:

$$h : \mathbf{x}_n \in \boldsymbol{X} \rightarrow h(\mathbf{x}_n) \in \{0, 1\} \quad (1)$$

En este trabajo nos centraremos en métodos de conjuntos (o *ensembles*) H_T tales que $H_T = \{h_t\}_{t=1}^{T}$. Los conjuntos están formados por predictores distintos generados a partir de un mismo conjunto de datos etiquetados. En el caso de que los predictores sean de distinto tipo, los conjuntos son heterogéneos. Los conjuntos compuestos de predictores del mismo tipo se denominan homogéneos. Aun siendo del mismo tipo, se pueden construir conjuntos de clasificadores distintos utilizando técnicas para generar diversidad. Un ejemplo de este tipo de técnicas es *bagging*. Se trata de un método cuyo fundamento se halla en la estadística no paramétrica. En concreto, encontramos una explicación profunda en (Efron & Tibshirani, 1993). En *bagging* se utilizan como conjuntos de entrenamiento muestras *bootstrap*, con o sin repetición del conjunto etiquetado original compuesto por *N* ejemplos. Cada uno de los T clasificadores del conjunto se entrena utilizando una muestra *bootstrap* independiente. La predicción final del conjunto se obtiene promediando las salidas de los integrantes del conjunto si el problema es de regresión, o utilizando voto por mayoría si el problema es de clasificación. Mediante este tipo de estrategias de diversificación y combinación de decisiones se espera mejorar la calidad de las predicciones.

Este trabajo de fin de grado tiene como propósito mejorar la eficacia y la eficiencia de conjuntos de clasificadores heterogéneos. Dicha propuesta se basa en agrupar los clasificadores mediante técnicas de *clustering*, basándonos en las predicciones que

obtengan sobre ciertos conjuntos de datos. En los siguientes apartados de la introducción se profundizará en la explicación de la propuesta y en la estructura de la memoria.

Llegados a este punto, se explicará el enfoque adoptado para ser capaces de agrupar clasificadores atendiendo a sus predicciones. Si se utilizan las técnicas de *bagging* descritas, se puede formar un conjunto de clasificadores de las características deseadas. De esta forma, en el momento de la predicción, obtendremos T clasificaciones distintas para un dado conjunto de test. Si interpretamos cada predicción como un punto de un espacio M-dimensional (recordemos que M es el número de ejemplos que integran $\boldsymbol{D}_{test}$), es fácil llegar a la conclusión de que se pueden agrupar las clasificaciones según algún criterio espacial, como por ejemplo la distancia existente entre ellas. Para este propósito, contamos con algoritmos de *clustering* o agrupamiento.

Como puntualización de la interpretación que hemos utilizado, hay que mencionar que, como las clases a las que pertenecen los ejemplos toman valores de un conjunto discreto (generalmente $\{0, 1\}$), el espacio M-dimensional descrito no será continuo y por lo tanto hay que ser cautelosos cuando se utiliza esta representación.

1.1 Motivación.

Actualmente, el aprendizaje automático se encuentra en un momento de auge. Son innumerables las aplicaciones que existen en todos los ámbitos de la vida personal y profesional.

Dado lo extenso del ámbito de las técnicas de aprendizaje automático, la mejora del rendimiento de éstas es un proyecto de investigación cuyo interés es claro. Al final, se pretende aumentar la efectividad de los sistemas utilizados para obtener mejores resultados en sus aplicaciones prácticas. En algunos casos, el buen funcionamiento de los métodos de aprendizaje automático es de importancia mayúscula: diagnosis médicas, predicciones de ventas, detección de ciberataques, etc. En otros ejemplos, el impacto causado puede ser de menor importancia, pero en cualquier caso se benefician en gran medida de cualquier aportación que consiga introducir una mejora en las técnicas que se pongan en práctica.

Es por eso que el presente trabajo centra su actividad en la búsqueda de un modo de reducir el tamaño de los grandes conjuntos de clasificadores utilizados actualmente a la vez que se trata de mejorar el error que cometen. Trataremos de sacar el mayor partido al entrenamiento de los distintos predictores combinándolos de forma inteligente. Como ya comentábamos, este agrupamiento se realizará aplicando algoritmos de *clustering* sobre las predicciones proporcionadas por los integrantes del conjunto.

No sólo se busca mejorar el rendimiento obtenido para las aplicaciones ya existentes, sino que también se pretende ampliar este espacio de posibilidades y descubrir nuevas utilidades o necesidades que puedan ser cubiertas por el aprendizaje automático. A pesar de que la motivación de este trabajo en concreto no va encaminada en esta línea, puede ser que surja la posibilidad durante el desarrollo de la misma.

1.2 Objetivos.

De acuerdo a las motivaciones expuestas en el apartado anterior, los objetivos de este trabajo de fin de grado son los siguientes:

- Diseñar una serie de funciones que nos permitan entrenar los clasificadores que elijamos mediante *bagging* para así generar conjuntos como los descritos.

- Implementar distintos algoritmos de *clustering* y comprobar cuál de ellos funciona mejor para alcanzar el objetivo propuesto. Así, elegiremos aquél que resulte más apropiado y evaluaremos su eficacia mediante un estudio empírico.

- A partir de la metodología implementada, realizar distintos experimentos para tratar de optimizar el uso de los grandes conjuntos heterogéneos que se utilizan actualmente para así mejorar el rendimiento de la clasificación que se obtiene en todas las posibles aplicaciones que se le puedan otorgar a dichos conjuntos.

- Evaluar los distintos resultados obtenidos para conseguir determinar si nuestras suposiciones iniciales se cumplen y podemos tomar la metodología desarrollada como un método válido para reducir el tamaño de un conjunto heterogéneo a la

vez que mejoramos el error de clasificación que resulta de la predicción global, o al menos, igualar aquél del conjunto completo.

- Comprobar que efectivamente se pueden agrupar los distintos clasificadores con respecto a sus predicciones si las entendemos como puntos de un espacio M-dimensional. Se considerará alcanzado este objetivo si se pueden formar clústeres de forma consistente y no aleatoria según el criterio definido. Con este propósito, se realizarán experimentos para discernir si el algoritmo de *clustering* escogido consigue separar las predicciones de los distintos clasificadores y agruparlas en torno a las de su misma procedencia.

- Profundizar en el análisis de los distintos clasificadores y su rendimiento. Comprobar la consistencia de sus resultados sobre los distintos conjuntos de datos.

A modo de resumen, el objetivo primordial del trabajo realizado es el de optimizar el uso de un conjunto heterogéneo de clasificadores. Para ello, realizaremos un estudio exhaustivo de las distintas posibilidades que nos ofrece el agrupamiento mediante *clustering* y las pruebas que nos permite hacer para hallar un rendimiento superior en términos de eficacia y eficiencia.

La eficacia mejorará en el caso en el que se consigan tasas de acierto mejores que las obtenidas por el conjunto completo. Esta posibilidad se estudiará mediante el agrupamiento de los clasificadores en clústeres. El objetivo principal del trabajo realizado es el de tratar de encontrar el punto en el cuál el conjunto obtiene el menor error sin llegar a añadir confusión a la predicción global. Este propósito está directamente relacionado con aquél de mejorar la eficiencia del conjunto. Si reducimos el tamaño del conjunto de manera inteligente, no sólo conseguiremos encontrarnos con un mejor error, sino que éste se conseguirá con menos recursos computacionales.

1.3 Organización de la memoria.

La memoria se organizará de la siguiente manera:

En el capítulo 2 se introducirá el tema principal del trabajo, proporcionando una visión general del estado actual de la tecnología. En este apartado, se abordará de manera somera el papel del aprendizaje automático en el mundo profesional con el fin de proporcionar un marco de referencia para los objetivos específicos de este trabajo. Se describirán trabajos previos en los que se abordan cuestiones relacionadas con esta investigación y que han sido utilizados como base para el desarrollo del presente documento.

Completado ya el comentario del contexto general del contenido que se desarrolla en este trabajo, entraremos en el capítulo 3, titulado "Diseño". Tratará sobre la propia explicación de la metodología que se ha implementado. Introduciremos la notación utilizada y seguidamente ahondaremos en la descripción de los algoritmos utilizados.

En el cuarto capítulo de la memoria se hablará de los detalles técnicos que han tomado parte en la implementación del proyecto: lenguaje de programación escogido, módulos implementados, etc.

Habiendo aclarado todo lo necesario para entender los procedimientos llevados a cabo, pasaremos al capítulo 5, que es la sección de los experimentos. Este apartado se dividirá en secciones que se corresponderán con cada experimento realizado. Cada una de estas secciones será introducida por una breve explicación del contexto en el que se desarrolla el experimento en cuestión y algunas aclaraciones que pueden resultar de interés.

En último lugar, llegaremos al quinto capítulo. A la luz de los resultados obtenidos en la sección anterior y sus respectivos análisis, se ofrecerán las conclusiones generales del trabajo al completo y se realizará una síntesis de los futuros caminos de actuación que se pueden tomar para completar el trabajo.

2. Estado del arte

Para entender el contexto en el que se enmarca este trabajo, es necesario dar una introducción al estado actual de la cuestión sobre la que trata. En este apartado, se analizará el papel del aprendizaje automático en la actualidad y se dará además una breve explicación de otros trabajos similares y las distintas aportaciones que han realizado a este campo.

2.1 Aprendizaje automático.

En la sociedad actual, existe una infinidad de aplicaciones directas del aprendizaje automático. Es bien sabido que este campo de la informática está experimentando *boom* a todos los niveles de la vida profesional.

En los últimos años, se han introducido nuevos conceptos que sirven para entender la utilidad del aprendizaje automático. Muchos artículos y presentaciones recientes hablan de lo que se conoce como "Internet de las Cosas" (*IoT; Internet of Things*). Los más destacados en la introducción de este concepto son (Ashton, 2009) y (Mattern & Floerkemeier, 2010). Esta nueva idea representa la visión del mundo en la cual el internet se introduce en la sociedad instalándose en los objetos más cotidianos para añadirles utilidad. De esta forma, se llega a una interconexión general en casi todos los aspectos de la vida ordinaria. Este constante enlace con la red permite a los dispositivos recopilar datos en todas partes y en todo momento. Por lo tanto, se abre un abanico amplísimo de posibilidades en cuanto a la utilidad que se le puede dar a estos datos. Es aquí donde el aprendizaje automático juega (y jugará cada vez más) un papel determinante. De la misma forma que se pueden recoger datos en cualquier circunstancia, se pueden utilizar para predecir casi cualquier cosa. El objetivo primordial es el de dotar a los dispositivos con autonomía para resolver problemas, tomar decisiones, predecir acontecimientos sin intervención humana, e incluso aprender de su propia experiencia. En definitiva, se trata de ofrecer una mayor utilidad al usuario sin necesidad de que éste participe.

Por esta misma razón, el aprendizaje automático está íntimamente relacionado con las técnicas de *Big Data*. La existencia de tal cantidad de datos necesita de nuevos métodos

que permitan manejarlos de manera eficiente y eficaz. Aunque no es un tema de verdadera trascendencia en el presente trabajo, es imperativo hablar del *Big Data* si se habla del estado del arte del aprendizaje automático.

Aunque sin duda el aprendizaje automático se encuentra en su momento de mayor utilización, ésta no hará sino crecer durante los próximos años. Decididamente, este campo, junto con muchos otros, revolucionará las distintas áreas tecnológicas y de negocio. Aunque los resultados ya se observan en la actualidad, su progresión va, indudablemente, en aumento.

2.2 Clustering de clasificadores.

Generalmente, el *clustering* se utiliza en el aprendizaje automático para labores de aprendizaje no supervisado. Es decir, en el cual no se conoce la clase a la que pertenece cada instancia del conjunto de datos. Un algoritmo de *clustering*, si se le proporciona el número de clases que existen, es capaz de agrupar los ejemplos de un conjunto dado en torno a los de su misma clase. Con esta solución, si bien no somos capaces de asignar etiquetas de clase, si nos es posible conocer qué ejemplos pertenecen a una misma clase.

Si bien las técnicas de *clustering* son ampliamente utilizadas en muchas aplicaciones del aprendizaje automático, el enfoque de este trabajo es menos habitual. Esta cuestión nos la encontramos de forma mucho menos frecuente en el ámbito profesional. Quizá sea porque se trata de una propuesta más profunda de lo que realmente se utiliza a día de hoy en el aprendizaje automático. Dicho de otra forma, la investigación en este campo no se trata de algo a partir de lo cual se obtenga un rendimiento inmediatamente perceptible, sino que sirve para tratar de mejorar el comportamiento de aquellos algoritmos que se utilizan en las aplicaciones prácticas que se les dan. De esta forma, el impacto que pueda tener un trabajo de experimentación se ve de forma menos directa.

Existen numerosas propuestas en la comunidad actual que utilizan de alguna manera técnicas de *clustering* sobre clasificadores. Las más habituales tratan de agrupar los clasificadores según los valores de sus parámetros. Existe una clara limitación en este tipo de trabajos, y es que solamente se puede trabajar sobre *conjuntos* de clasificadores

homogéneos. Esto es así porque muy pocos algoritmos pueden ser comparados entre sí mediante sus parámetros, pues ni son iguales, ni pueden tomar los mismos valores ni por supuesto cumplen el mismo papel en la clasificación. Un ejemplo muy claro de este enfoque lo encontramos en (Bakker & Heskes, 2003), donde se estudia el *clustering* de redes neuronales en el espacio de los pesos sinápticos asignados a cada neurona.

No obstante, si queremos ser capaces de aplicar *clustering* sobre conjuntos heterogéneos, tenemos que llevar a los distintos clasificadores a un espacio en el que todos sean comparables. Evidentemente, el único espacio en el cual todos ellos pueden ser comparados es el espacio de predicciones, pues para cada conjunto de datos, cualquier predicción obtenida por cualquier clasificador va a ser similar en forma y estructura. De este modo, todas las predicciones se pueden situar en el mismo espacio para así ser comparadas. El ejemplo más apropiado del agrupamiento de clasificadores en el espacio de predicciones lo encontramos, igualmente, en (Bakker & Heskes, 2003). El algoritmo propuesto en este artículo ha resultado ser más apropiado que los ya conocidos para esta tarea.

3. Diseño

A continuación se especifican los métodos de *clustering* que se han utilizado para identificar los conglomerados: K-Medias difuso (*fuzzy K-means)* y *clustering* mediante templado determinista (*deterministic annealing).* Encontramos explicaciones completas de estos algoritmos en (Bezdek, et al., 1984) y (Bakker & Heskes, 2003), respectivamente. En la descripción de estos algoritmos, supondremos las siguientes condiciones:

Sea T el número total de clasificadores del conjunto. Cada uno de los clasificadores del conjunto está caracterizado caracterizado por su vector de predicciones $\boldsymbol{y}_t$, $(1 \leq t \leq T)$ en un conjunto de tamaño M. Inicialmente, de manera aleatoria, escogemos un número K de vectores de predicción de entre todas las T del conjunto. Estos K vectores de predicción actuarán como centroides iniciales de los clústeres que queremos formar. Cada uno de estos centroides será denotado por $\boldsymbol{m}_k$, $(1 \leq k \leq K)$. De esta forma, usaremos $D(\boldsymbol{y}_t, \boldsymbol{m}_k)$ para referirnos a la distancia que hay desde el clasificador t hasta el centroide k. Para calcular $D(\boldsymbol{y}_t, \boldsymbol{m}_k)$ se puede utilizar cualquier función de distancia apropiada. Generalmente utilizaremos la distancia euclidea, pero podrían ser utilizados el error cuadrático medio, el error de entropía cruzada, etc.

3.1 Fuzzy K-Means.

Este método de *clustering* es quizá el más conocido. Parte de su versión menos refinada (*K-Means;* (MacQueen, 1967)) para optimizar su rendimiento. En el algoritmo *K-Means* los puntos pertenecen exclusivamente al clúster cuyo centroide se encuentra a una menor distancia.

En cambio, en *fuzzy K-Means*, se introduce el concepto de grado de pertenencia a un clúster: todos los puntos pertenecen a todos los clústeres en cierto grado. El grado de pertenencia de un punto a un clúster cuantifica la "fuerza" con la que atrae el centro del

clúster a ese punto. Para la asignación final, un ejemplo dado es asignado al clúster para el cual el grado de pertenencia es mayor.

Sea u_{ik} el grado en el que el punto *t* pertenece al clúster k:

$$u_{tk} = \frac{1}{\sum_{k'} \frac{D(\boldsymbol{y}_t, \boldsymbol{m}_k)}{D(\boldsymbol{y}_t, \boldsymbol{m}_{k'})}} \tag{2}$$

De esta forma, si *K-Means* asigna el punto *t* al clúster k que minimice $D(\boldsymbol{y}_t, \boldsymbol{m}_k)$, *fuzzy K-Means* lo hará a aquel que maximice u_{tk}. Entonces, el paso de maximización de este algoritmo buscará actualizar el centroide $\boldsymbol{m}_k$ para hallar un centroide $\boldsymbol{m}'_k$ tal que:

$$\boldsymbol{m}'_k = \frac{\sum_t \boldsymbol{y}_t u_{tk}^2}{\sum_t u_{tk}^2} \tag{3}$$

Dicha actualización se llevará a cabo como último paso de cada iteración hasta que $D(\boldsymbol{m}_k, \boldsymbol{m}'_k) < \varepsilon$, donde ε es un valor suficientemente pequeño (típicamente del orden de 10^{-6}).

3.2 Clustering por "deterministic annealing".

Ésta técnica de *clustering* no es sino una adaptación del algoritmo de *fuzzy K-Means* para mejorar su rendimiento cuando se aplica sobre predicciones en lugar de puntos de un espacio continuo.

El fundamento de este algoritmo de *clustering* fue propuesto en (Bakker & Heskes, 2003). Partiendo de la interpretación del grado de pertenencia como fuerza de atracción, este algoritmo utiliza una analogía física haciendo uso del concepto de energía libre o energía de Gibbs para definir los grados de pertenencia. Así, la derivación de esta energía nos permitirá hallar su máximo de forma sencilla para utilizarlo en el paso de maximización.

3.2.1 La derivación de la energía libre.

La energía libre como potencial termodinámico se enuncia como:

$$G = H - TS, \tag{4}$$

siendo *H* la energía cedida o absorbida por el sistema (que tomaremos como la energía de cada centroide con cada predicción), *T* la temperatura y *S* la entropía. Si bien esta enunciación no revela por sí sola el fundamento del enfoque propuesto en (Bakker & Heskes, 2003), todas las aclaraciones de cómo interpretaremos esta relación para adecuarlo a nuestro propósito se verán a continuación.

Utilizaremos p_{tk} para referirnos a la probabilidad de que el clasificador t pertenezca al clúster k. Para calcular esta probabilidad, utilizamos una función *softmax* dependiente del centroide escogido y la distancia del mismo al punto en cuestión:

$$p_{tk}(\boldsymbol{m}) = \frac{e^{-\beta D(y_t, \boldsymbol{m}_k)}}{\sum_{k'} e^{-\beta D(y_t, \boldsymbol{m}_{k'})}}, \tag{5}$$

donde β es el inverso de la temperatura (Rose, et al., 1990), y $\sum_k p_{tk} = 1 \; \forall t$. Este es el parámetro que utilizaremos para *deterministic annealing*. Para valores de β próximos a 0 (temperatura elevada, que son las condiciones que se utilizarán al principio del proceso de templado), $p_{tk} = p_{t'k'} \; \forall \, t, k, t', k'$, es decir los puntos pertenecen por igual a todos los clústeres. Para valores grandes de β (temperatura baja, como en las fases finales del templado), la probabilidades de pertenencia se concentran en el clúster más próximo.

El objetivo del algoritmo de *clustering* es converger a un punto en el cual los centroides existentes minimicen la distancia media que hay hasta las predicciones que agrupamos. Dada la definición anterior, tenemos que para este objetivo debemos minimizar una energía media que calculamos de esta forma:

$$E(\boldsymbol{M}, \boldsymbol{P}) = \sum_{tk} p_{tk}\, D(\boldsymbol{y}_t, \boldsymbol{m}_k), \tag{6}$$

donde $\boldsymbol{M}$ y $\boldsymbol{P}$ son todos los centroides $\boldsymbol{m}_k$ y las probabilidades p_{tk}. De esta forma, la energía media es la media de las distancias de los puntos a los centroides ponderada por p_{tk}.

Para llegar hasta la energía libre que queremos maximizar, definimos una entropía $S(\boldsymbol{P})$, que en el momento de maximizarla, va a favorecer el estado de caos necesario para partir de una situación en la que no exista conocimiento previo alguno acerca de la estructura de los clústeres. Utilizaremos la versión discreta de la entropía de Shannon:

$$S(P) = -\sum_{tk} p_{tk} \log p_{tk} \tag{7}$$

Introducir esta entropía en la función nos ayuda a solucionar el siguiente problema: dados los todos los centroides $\boldsymbol{m_k}$, si minimizamos la energía media no estaríamos sino asignando cada predicción al clúster cuyo centroide se encuentra a una distancia menor con probabilidad total. Esta solución sería equivalente a todos los efectos a lo que obtuviera *K-Means* bajo las mismas circunstancias.

Añadiendo el término que denominamos "parámetro de regularización" (correspondiente a *T* en (4)) llegamos a la "energía libre":

$$F(\boldsymbol{M}, \boldsymbol{P}) = E(\boldsymbol{M}, \boldsymbol{P}) - TS(\boldsymbol{P}) \tag{8}$$

Tomando $\boldsymbol{P}$ como el conjunto de todas las probabilidades para cada centroide $\boldsymbol{m}$ ($\boldsymbol{P} = \{\boldsymbol{p}(\{\boldsymbol{m}\})\}$), al sustituir en la energía libre como vemos en (Bakker & Heskes, 2003), llegamos a:

$$F(\boldsymbol{M}, \boldsymbol{P}) = \sum_{t} \log \sum_{k} e^{-\beta D(\boldsymbol{y}_t, \boldsymbol{m}_k)} \tag{9}$$

Como se menciona en el artículo, este resultado es equivalente al propuesto en otros artículos como (Buhmann & Kühnel, 1993) o (Rose, et al., 1990) con ciertos matices de menor importancia.

3.2.2 Paso de maximización.

Como ya ha sido introducido, el algoritmo de *deterministic annealing* comenzará utilizando valores de β pequeños para posteriormente aumentar dicho parámetro con el avance de las iteraciones.

Avanzando ya a hablar del paso de maximización del algoritmo, el objetivo es encontrar, para cada valor de β (equivalentemente, en cada iteración), centroides para cada clúster minimizando a energía libre explicada en el apartado anterior. Esta minimización se lleva a cabo resolviendo un sistema formado por una ecuación para cada $\boldsymbol{m_k}$ que será de la forma:

$$\frac{\delta F}{\delta \boldsymbol{m}_k} = \sum_t p_{tk} \frac{\delta D_{tk}}{\delta \boldsymbol{m}_k} = 0 \tag{10}$$

La solución de este sistema es hallada siguiendo un algoritmo de *Expectation-Maximization (EM algorithm)* que encontramos en (Rubin, 1991).

De esta manera, en el paso de maximización, se busca actualizar los centroides anteriores de forma que el centroide resultante $\boldsymbol{m}'_k$ maximice $\sum_t p_{tk} D(\boldsymbol{y}_t, \boldsymbol{m}_k)$. Así, enunciamos formalmente:

$$\boldsymbol{m}'_k = \underset{\boldsymbol{m}_k}{\operatorname{argmax}} \sum_t p_{tk} D(\boldsymbol{y}_t, \boldsymbol{m}_k) \tag{11}$$

Hasta este punto, la explicación del algoritmo equivale a la que nos encontramos en (Bakker & Heskes, 2003). La principal diferencia es que, en vez de maximizar (11) mediante un algoritmo de descenso por gradiente, podemos hacerlo derivando la propia expresión si elegimos de forma apropiada la función de distancia que utilizamos. Este paso requiere de una explicación detallada que no se proporciona en el mencionado artículo. Existe una errata en el documento, pues el resultado correcto para esta maximización no es el que encontramos expuesto en él. Veremos a continuación la demostración.

La explicación que sigue corresponde a la manera de hallar el máximo de (11). Como ya introducíamos antes, si elegimos de forma inteligente función para calcular la distancia, obtenemos una manera realmente sencilla de hallar dicho máximo. Sin embargo, la explicación de cómo llegamos a ese punto es necesaria.

Tanto si elegimos *sum-squared error* como *cross-entropy error* para calcular la distancia, el resultado es exactamente el mismo.

Partimos de la definición de distancia según la función *sum-squared error*:

$$D(\boldsymbol{y}_t, \boldsymbol{m}_k) = \|\boldsymbol{y}_t - \boldsymbol{m}_k\|^2 = (\boldsymbol{y}_t - \boldsymbol{m}_k)^T \cdot (\boldsymbol{y}_t - \boldsymbol{m}_k) \tag{12}$$

Para nuestro propósito, incluimos (12) en (10) de tal forma que obtengamos:

$$\begin{aligned} &\frac{\delta}{\delta \boldsymbol{m}_k} \sum_t p_{tk} (\boldsymbol{y}_t - \boldsymbol{m}_k)^T \cdot (\boldsymbol{y}_t - \boldsymbol{m}_k) \\ &= -2 \sum_t p_{tk} (\boldsymbol{y}_t - \boldsymbol{m}_k) \end{aligned} \tag{13}$$

Para hallar el máximo, igualamos el resultado de (13) a 0 (condición de máximo) y continuamos operando para despejar $\boldsymbol{m}_k$:

$$\sum_t p_{tk} (\boldsymbol{m}_k - \boldsymbol{y}_t) = 0;$$

$$\boldsymbol{m}_k \sum_t p_{tk} - \sum_t p_{tk} \boldsymbol{y}_t = 0; \tag{14}$$

$$\boxed{\boldsymbol{m}_k = \frac{\sum_t p_{tk} \boldsymbol{y}_t}{\sum_t p_{tk}}}$$

Como ya había sido anticipado, llegamos a una expresión realmente simple para hallar los nuevos centroides. El paso de maximización será, sencillamente, actualizar los

centroides $\boldsymbol{m}_k'$ según el resultado hallado para (14). Podemos encontrarnos con el mismo resultado si utilizamos la función de distancia de *cross-entropy error*. La derivación de esta función es algo más complicada, y dado que no es necesaria, no la incluiremos en esta memoria.

La principal diferencia que nos encontramos con (Bakker & Heskes, 2003) es que el resultado al que llegan no está normalizado por el sumatorio de toda probabilidad de cada punto t de pertenecer al clúster k. Una posible explicación es que se podría llegar a pensar que $\sum_t p_{tk} = 1 \; \forall k$, y que por tanto nos podríamos olvidar del denominador en el resultado de (14). Sin embargo, esto es falso. La realidad es que $\sum_k p_{tk} = 1 \; \forall t$, pero esto no nos permite obviar esa parte de (14). De hecho, no nos aporta nada relevante en este caso.

3.2.3 Detalles concretos del algoritmo.

Antes de revelar los valores elegidos para β, nótese que con $\beta = 0$, cada punto es asociado a todos los clústeres con la misma probabilidad. A medida que β crece, las asociaciones a los clústeres se vuelven menos difusas (en inglés, *fuzzy*). Cuando $\beta \to \infty$, cada punto pertenece a un solo clúster con probabilidad 1.

Como ya se ha señalado anteriormente, comenzaremos con valores de β pequeños y crecerán con el paso de las iteraciones. Para ello, evidentemente, hay que definir dos valores antes de proceder a la experimentación. El primero de ellos es el valor inicial que tendrá este parámetro. Si utilizamos $\beta = 1$, partiremos de una situación de "igualdad de condiciones" en todos los casos, por lo que parece un valor inicial apropiado.

El incremento de β (en adelante $\Delta\beta$), por su parte, es menos fácil de elegir. La realidad es que no ha causado un impacto verdaderamente determinante en el resultado final, pero siendo rigurosos es evidente que hay que elegirlo con cuidado. Para $\Delta\beta = 1$, encontramos resultados satisfactorios en todos los casos que hemos probado, por lo que partiremos de este valor para nuestras experimentaciones.

4. Detalles de implementación

Al tratarse de un proyecto de ingeniería de software, no podemos ofrecer simplemente una explicación del fundamento teórico del trabajo que se ha realizado. Se debe también prestar mucha atención a los aspectos técnicos que se han considerado a la hora de la implementación, así como a dar una explicación de los mismos al lector.

4.1 Lenguaje de programación escogido.

El principal considerando de la codificación es el lenguaje de programación que se ha utilizado para la misma. Por razones que explicaremos a continuación, se ha elegido *Python*. Se trata de un lenguaje de muy alto nivel cuyo código resulta, generalmente, fácilmente legible e interpretable. Dada su versatilidad, soporta paradigmas de programación complementarios, como la orientación a objetos o la programación funcional. Además, puede ser utilizado para programación científica, aplicaciones web, aplicaciones de escritorio, etc. Para la utilización de este lenguaje, ha resultado de gran utilidad la documentación proporcionada al público por los desarrolladores que encontramos en (Python Software Foundation, 2016), pero también otros sitios web no oficiales como (López Briega, 2015).

Esta elección viene respaldada por el altísimo grado de adaptación que tiene este lenguaje al desarrollo de trabajos de investigación científica, ingeniería, etc. Adicionalmente, incluye ciertas herramientas muy útiles para trabajar con métodos de aprendizaje automático.

Las librerías *NumPy* y *SciPy* proporcionan una funcionalidad crucial en a cualquier proyecto de computación científica. En particular, *NumPy* incluye numerosas utilidades que permiten trabajar con vectores y matrices, además de herramientas matemáticas de alto nivel. *SciPy*, por su parte, ofrece incontables funciones matemáticas de gran utilidad científica.

Además, *Python* cuenta con el paquete *Matplotlib*. Se trata de un entorno que ofrece una amplia oferta de posibilidades a la hora de representar datos en gráficas 2D. Por razones evidentes, nuestro proyecto requiere de una manera útil y versátil de obtener estas representaciones gráficas para mostrar los resultados de la experimentación de forma clara e intuitiva, motivo por el cual necesitamos esta herramienta.

Como último argumento (quizá el más importante) que respalda nuestra elección del lenguaje, comentaremos la gran utilidad de la librería *scikit-learn* (Pedregosa, et al., 2011) con la que cuenta *Python*. Esta es quizá la herramienta más completa que podemos encontrar para usar algoritmos y técnicas de aprendizaje automático. Cuenta con numerosas funciones dedicadas a la minería y análisis de datos. Como detalle especialmente útil que requerimos en nuestro trabajo, cabe destacar la implementación de innumerables algoritmos de clasificación que existe en esta librería a disposición del que quiera utilizarlas.

4.2 Metodología del proyecto.

En este apartado, explicaremos de forma resumida todos los métodos implementados para desarrollar el trabajo.

En primer lugar, comentar que han sido programados varios módulos o bibliotecas que contienen las funciones necesarias para llevar a cabo la experimentación. Además, para cada tipo de experimento, se ha creado un *script* que realiza la labor de función principal, el cual se encarga de llamar a las funciones de las librerías mencionadas para así obtener resultados y representarlos.

El módulo principal que se ha utilizado contiene las implementaciones de los algoritmos de *clustering* descritos en el apartado 3, además de numerosas funciones auxiliares que desarrollan tareas necesarias tales como cuantificar la tasa de impureza de los clústeres, calcular el error máximo que se puede cometer dadas unas circunstancias cualesquiera o hallar la distancia entre puntos. Esta biblioteca se encuentra en el archivo *clustering.py*, del cual se incluirán los fragmentos más interesantes en la sección de anexos en forma de pseudocódigo.

Otro módulo importante es el que comprende toda la funcionalidad requerida para el entrenamiento y la clasificación de los predictores. En esta librería, las funciones más utilizadas se encargan de los distintos tipos de validación: simple, cruzada, *bagging* con y sin repetición, etc. Por supuesto, también contiene funciones de utilidad auxiliar: calcular el error de clasificación cometido, obtener la predicción de un conjunto usando voto por mayoría, etc. El archivo que contiene este módulo es *clasificacion.py*.

Aunque su papel sea complementario, también merecen ser mencionadas las librerías usadas para graficar los resultados (*graficas.py*), disponer los conjuntos de datos según el formato adecuado (*datos.py*) y generar utilidades para probar el funcionamiento de los algoritmos (*puntos.py*, *matrices.py*).

Por último, los *scripts* utilizados para los experimentos son *curvas_clustering.py*, *subbagging.py* y *mejor_cluster.py*.

A modo de aclaración importante, todas las funciones que realizan una labor equivalente (por ejemplo, los algoritmos de *fuzzy K-Means* y de (Bakker & Heskes, 2003)) cuentan con los mismos parámetros en su cabecera. Los argumentos que lo permiten, tienen valor asignado por defecto para que no sea necesario especificarlos todos en todo momento. De esta forma, se puede experimentar muy cómodamente sin necesidad de conocer la estructura interna de ningún algoritmo.

Además de los algoritmos de *clustering*, también se incluirá en la sección de anexos cualquier pseudocódigo que pueda resultar de interés o que pueda no haber quedado suficientemente claro con la explicación que se le ha dado.

5. Validación empírica del método

En esta sección presentaremos los resultados de un análisis empírico del método diseñado en este trabajo, cuyo objetivo es identificar agrupaciones de clasificadores dentro de un conjunto cuya capacidad de generalización sea comparable o superior a la del conjunto completo. Se busca de esta forma mejorar las cualidades de eficacia y eficiencia de los conjuntos de clasificadores. Para determinar el éxito del proyecto se han realizado distintos tipos de experimentos en problemas de clasificación de diversas áreas de aplicación. Dichos problemas han sido obtenidos del repositorio web (Bache & Lichman, s.f.).

El capítulo está dividida en tres secciones, en cada una de las cuales se presentan los resultados de un grupo de experimentos. El objetivo del primer bloque de experimentos es analizar el desempeño y el correcto funcionamiento de los algoritmos diseñados, para así poder determinar cuál resulta más apropiado para nuestro propósito. En el segundo, se realiza un análisis más profundo del algoritmo que hemos escogido dada la información obtenida en el primer experimento. El tercero es quizá el más importante, ya que su objetivo es determinar la utilidad práctica del método propuesto.

5.1 Clasificadores utilizados.

Para los experimentos utilizaremos tres tipos de clasificadores.

- **Perceptrón multicapa (*Multilayer perceptron*).** Se trata de una red neuronal artificial que cuenta con múltiples capas de neuronas, lo que le otorga la peculiaridad frente al perceptrón simple de poder resolver problemas que no son linealmente separables. Hemos decidido utilizar una única capa oculta con 10 neuronas, ya que ha resultado ofrecer un buen rendimiento en general con estos parámetros.

- **Bosque aleatorio (*Random forest*).** Este algoritmo, como se explica extensamente en (Breiman, 2001), el artículo que lo introduce originalmente, se compone de un conjunto de árboles clasificadores, por lo que realmente podemos tomarlo como un conjunto de clasificadores. A la hora de obtener predicciones, los distintos árboles clasifican el ejemplo en cuestión y luego se realiza una "puesta en común" de la cual sale la clasificación final del algoritmo utilizando, generalmente, un voto por mayoría.

- **Árbol de decisión (*CART*).** Este último es el que utilizaremos para profundizar en el estudio de los propios algoritmos. Se trata de un árbol de decisión simple que se construye dividiendo cada uno de sus nodos en los hijos pertinentes de acuerdo a varios criterios que miden el impacto de cada atributo y sus valores. La exposición completa de este clasificador puede encontrarse en el libro (Breiman, et al., 1984).

Los dos primeros tienen interés especial, pues los utilizaremos en todos los experimentos. El tercero ha sido incluido para profundizar en el análisis del comportamiento de los algoritmos y así determinar cuál de ellos ofrece un mejor desempeño cuando las circunstancias se complican (se incluyen más clasificadores o se utilizan conjuntos de datos más complejos).

5.2 Conjuntos de datos utilizados.

Los conjuntos de datos seleccionados para el estudio son de diversa naturaleza, en cuanto a tamaño y número de atributos. Se trata de problemas de clasificación binaria y ternaria obtenidos de (Bache & Lichman, s.f.). Algunos han sido utilizados únicamente en algún grupo experimentos. La razón de esta exclusión ha sido que, para observar los resultados de dichos experimentos, no ha sido necesario utilizar tantos, ya que las gráficas que obtenemos son muy similares en todos los casos.

Nombre del conjunto	*Tamaño*	*Clases*	*Atributos*
Diabetes (Pima)	768	2	8
German	1000	2	20
Breast cancer Winconsin	569	2	30
Tic-tac-toe	958	2	9
Blood	748	2	4
Heart	303	3	13
Liver (BUPA)	345	2	6
Chess	3196	2	36
SPECT Heart	267	2	22
Cars	1728	2	6

Tabla 5-1. Conjuntos de datos utilizados

5.3 Evolución del grado de impureza.

En este primer bloque de experimentos se tratará de determinar si el algoritmo propuesto en (Bakker & Heskes, 2003) se adecúa a nuestro objetivo mejor que *fuzzy K-Means*.

Para ello, se ha monitorizado la evolución del grado de impureza de los clústeres identificados por los distintos algoritmos en función de las iteraciones del algoritmo de *clustering* realizadas.

El grado de impureza cuantifica cuán heterogéneos son los clústeres obtenidos. Se define como el porcentaje de clasificadores que pertenecen a un clúster compuesto en su mayoría por clasificadores de otro tipo. Por ejemplo, la asignación de un árbol *CART* a un clúster formado mayoritariamente por redes neuronales, contribuiría a incrementar la impureza del clúster. Observemos que esta asignación no tiene por qué conducir a una menor precisión en las predicciones. Simplemente es consecuencia de que el clasificador, a pesar de ser un árbol de decisión, predice de manera similar a las redes neuronales asignadas a ese clúster.

El protocolo empírico es el siguiente: partimos de un conjunto de T clasificadores de cada tipo generados mediante *bagging*. Si tomamos N tipos de clasificadores distintos, tenemos NT predictores. Estos NT predictores son utilizados como puntos para realizar *clustering*. La dimensión de cada uno de estos puntos vendrá definida por el tamaño del conjunto de ejemplos que se utilice para obtener las predicciones. En los experimentos realizados ha sido usado el conjunto de test.

Atendiendo a la definición dada anteriormente para el grado de impureza de los clústeres encontrados, el número máximo de errores que pueden darse en una iteración del algoritmo es:

$$\frac{N-1}{N} NT = (N-1)T \tag{15}$$

Esta cantidad ha sido utilizada para normalizar el grado de impureza de forma que sea un porcentaje y se puedan comparar el desempeño de distintos métodos.

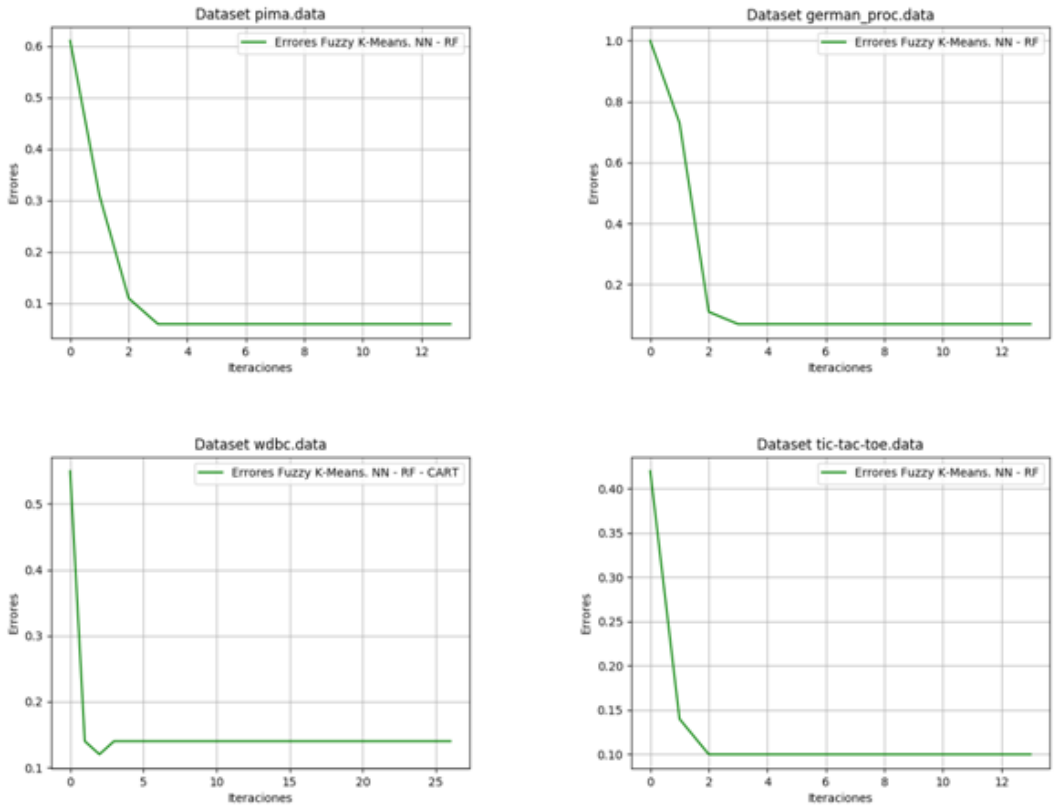

Fig 5-1. Grados de impureza para los clústeres identificados por fuzzy K-Means frente al número de iteraciones para conjuntos formados por perceptrones multicapa y árboles aleatorios

En esta primera figura observamos los resultados del experimento explicado utilizando el algoritmo *fuzzy K-Means* sobre un conjunto de predicciones obtenidas por T perceptrones multicapa y T árboles aleatorios. En cada figura, se utiliza como título el nombre del conjunto de datos utilizado. Para estos primeros pasos de la experimentación, se presentan los resultados para cuatro conjuntos representativos.

Las curvas de error para todos problemas de clasificación presentan características muy similares: durante las primeras iteraciones del algoritmo, el error cometido es bastante elevado. Esto significa que un porcentaje elevado de los NT clasificadores han sido asignados a clústeres en los que la mayoría de clasificadores son de distinto tipo. Teniendo en cuenta que el inicio del algoritmo depende en gran medida de la elección azarosa de los centroides, este resultado es esperable. Sin embargo, al cabo de pocas iteraciones, el algoritmo consigue agrupar los clasificadores en clústeres muy homogéneos, en algunos

casos casi puros. A partir de este momento se modera la velocidad de descenso de la curvas de error y los algoritmos convergen.

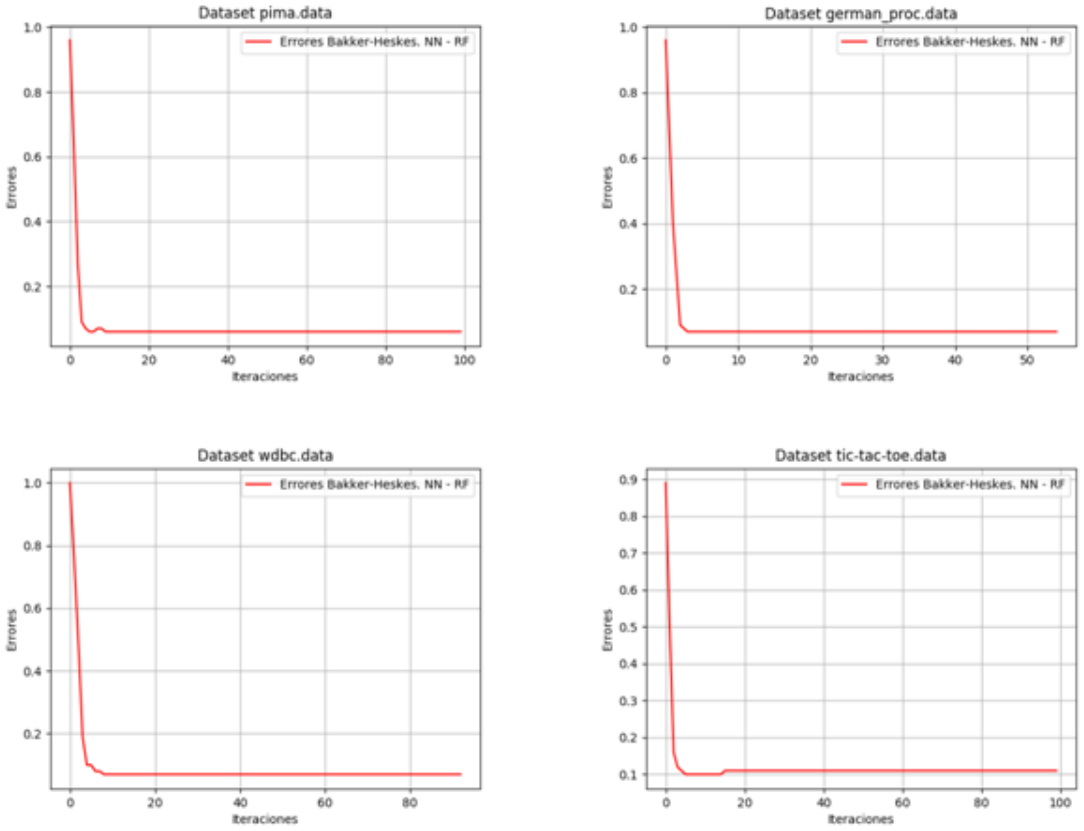

Fig 5-2. Grados de impureza para los clústeres identificados por Bakker-Heskes frente al número de iteraciones para conjuntos formados por perceptrones multicapa y árboles aleatorios

En esta segunda gráfica vemos los resultados de ejecutar el algoritmo propuesto en (Bakker & Heskes, 2003) bajo las mismas circunstancias que las utilizadas para obtener la figura 1.

Observamos un comportamiento muy similar al anterior. El algoritmo converge al cabo de pocas iteraciones, y conduce a clústeres bastante homogéneos. Como principales diferencias, hay que señalar dos: La primera es que el error inicial medio en este caso parece ser bastante mayor (casi el 100%). La segunda y más importante es que este algoritmo necesita más iteraciones para converger.

Tras analizar los resultados obtenidos, pudiera parecer que el algoritmo de (Bakker & Heskes, 2003) es inferior a *fuzzy K-Means* para alcanzar la meta que hemos establecido: el error final de *clustering* es muy similar en ambos casos, Sin embargo *fuzzy K-Means* converge más rápido que el algoritmo propuesto por (Bakker & Heskes, 2003). Sin embargo, con la información de la que disponemos hasta el momento, que es incompleta, no podemos asegurar que esta conclusión sea válida.

Probaremos a continuación que ocurre si introducimos un tercer tipo de clasificador. De acuerdo con los experimentos exploratorios que hemos realizado, el tipo de algoritmo de clasificación elegido parece no ser determinante para el comportamiento que se observa en conjuntos con tres tipos de clasificadores. Si bien se han realizado experimentos con diferentes tipos de clasificadores, incluiremos solamente los resultados con conjuntos que, además de redes neuronales y árboles aleatorios, incluyen árboles de decisión *CART* (*decision tree*), pues estos parecen suficientemente ilustrativos y representativos.

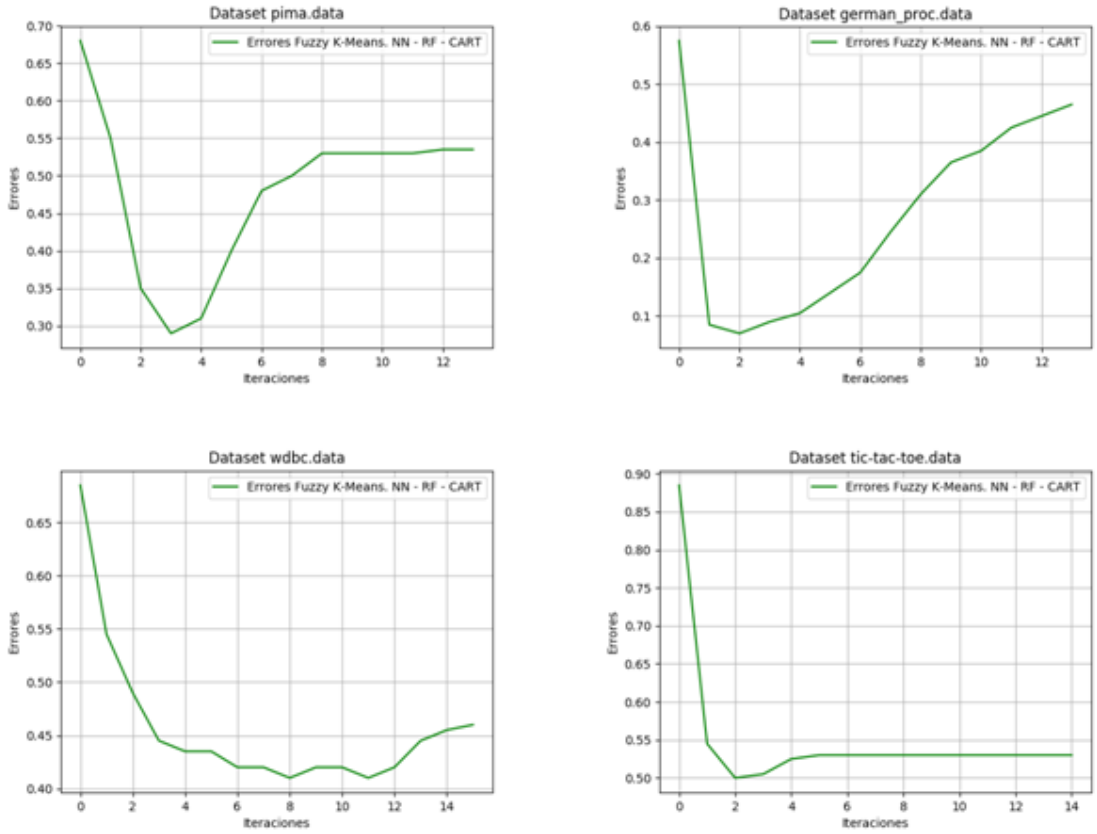

Fig 5-3. Grados de impureza para los clústeres identificados por fuzzy K-Means frente al número de iteraciones para conjuntos formados por perceptrones multicapa, árboles aleatorios y árboles de decisión

Como se puede apreciar claramente en las gráficas de la figura 3, el comportamiento de *fuzzy K-Means* es menos estable que antes. La tendencia que observamos ya no es común en todas las gráficas. Tampoco es igual a la que se obtiene cuando los conjuntos están formados solamente por dos tipos de clasificadores (redes neuronales y árboles aleatorios). Adicionalmente, los clústeres finales son menos homogéneos que en el experimento previo. En todos los casos el error de *clustering* se aproxima al 50%.

Además, las diferencias entre las curvas de error obtenidas en distintos problemas de clasificación sugieren que *fuzzy K-Means* no es una buena estrategia para realizar *clustering* entre los clasificadores de un conjunto basado en sus predicciones. Por lo tanto, hemos preferido en este trabajo utilizar el algoritmo propuesto por (Bakker & Heskes, 2003), cuyo comportamiento parece ser más estable. En parte esta decisión se justifica por el interés de incluir en los conjuntos más tipos de clasificadores como trabajo futuro.

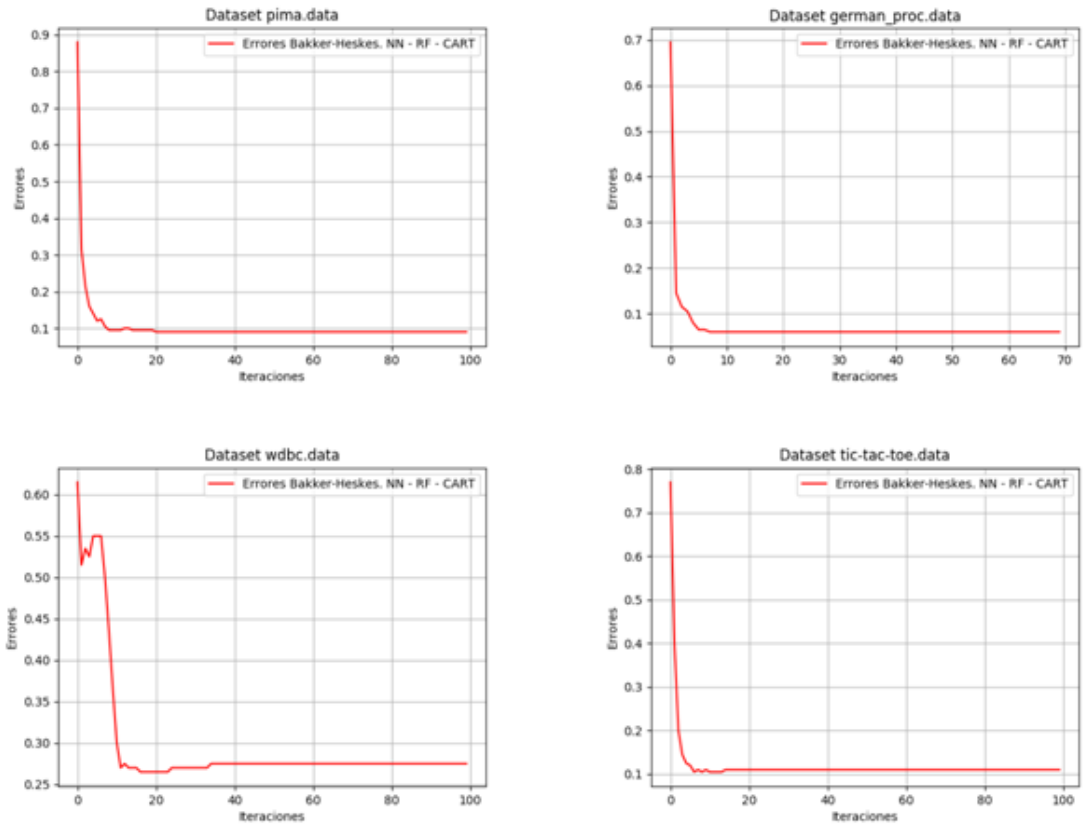

Fig 5-4. Grados de impureza para los clústeres identificados por Bakker-Heskes frente al número de iteraciones para conjuntos formados por perceptrones multicapa, árboles aleatorios y árboles de decisión

Para terminar con este apartado de la investigación, mostramos la evolución del algoritmo de (Bakker & Heskes, 2003) ejecutado bajo las mismas condiciones que los experimentos anteriores.

Es notable que el resultado obtenido en este caso, si bien menos uniforme que el de los primeros experimentos de este algoritmo, es mejor que el anterior.

Respecto a la eficacia de este algoritmo, nos encontramos con un error final considerablemente menor que el que conseguido por *fuzzy K-Means*. La tendencia de las gráficas, sin embargo, si se muestra más irregular en esta segunda parte de la experimentación, aunque también parece más estable que la de *fuzzy K-Means*.

Llegados a este punto, podemos considerar que tenemos suficiente información para establecer que, para la tarea que nos interesa, el algoritmo de (Bakker & Heskes, 2003) es más apropiado que *fuzzy K-Means*.

Habiendo llegado a esta conclusión, los siguientes pasos de la experimentación se realizarán únicamente con el algoritmo que hemos escogido como mejor.

5.4 Subbagging: análisis de resultados en función del tamaño del conjunto de entrenamiento.

Subbagging es una variante de *bagging* en la que los conjuntos de entrenamiento que se generan por remuestreo no necesariamente tienen el mismo tamaño que el original. En esta sección analizaremos el rendimiento del algoritmo propuesto en función del tamaño de dichos conjuntos.

La idea a partir de la cual surge esta experimentación es la siguiente. Es razonable pensar que cuanto menor sea el conjunto de entrenamiento utilizado, menor será la información de la que disponen los clasificadores individuales sobre los patrones de dependencia entre los atributos y las etiquetas que permiten hacer predicciones. Por lo tanto, es de esperar que la varianza de dichas predicciones sea mayor que cuando se entrena con tamaños mayores. Por esta misma razón, cabe esperar que, para tamaños menores del conjunto de entrenamiento la impureza de los clústeres sea mayor.

Para concretar la idea explicada, realizaremos una serie de experimentos en los que haremos variar el tamaño del conjunto de entrenamiento usado para entrenar los clasificadores individuales en *bagging* entre un 10% y el 100%. Nótese que, como utilizamos remuestreo con repetición para seleccionar los ejemplos del conjunto de entrenamiento, puede existir copias múltiples de algunos ejemplos en el conjunto de entrenamiento final, por lo es posible seguir aumentando el porcentaje por encima del 100%. No obstante, no hemos incluido en esta memoria los resultados de experimentos con tasas de remuestreo por encima de este valor, ya que las conclusiones del análisis realizado se perfilan claramente antes de alcanzar este porcentaje.

Si con los experimentos anteriores conseguimos determinar que algoritmo se adecuaba mejor a nuestro objetivo, con esta segunda fase de la experimentación conseguiremos averiguar cuál es el tamaño del conjunto de entrenamiento que resulta óptimo para observar los resultados que buscamos. Para ser aceptado, este porcentaje deberá ser persistente en las distintas partes de estos experimentos. Por lo tanto, utilizaremos aquél porcentaje que sea menor de entre todos con los que se consiga obtener un resultado óptimo.

Debido a la naturaleza aleatoria del algoritmo que utilizamos para generar los conjuntos es necesario realizar varias ejecuciones sobre los mismos datos con el fin de obtener conclusiones fiables. Los resultados de estos experimentos se presentan en un diagrama de caja mediante el que se representan no solo los errores medios obtenidos, sino también la dispersión de valores para cada tamaño del conjunto de entrenamiento considerado.

De igual manera que con los experimentos anteriores, no sólo nos basta probar el algoritmo para las predicciones de los clasificadores que usaremos más adelante, sino que haremos pruebas más allá para asegurarnos que los resultados sugeridos son persistentes cuando se cambian las condiciones.

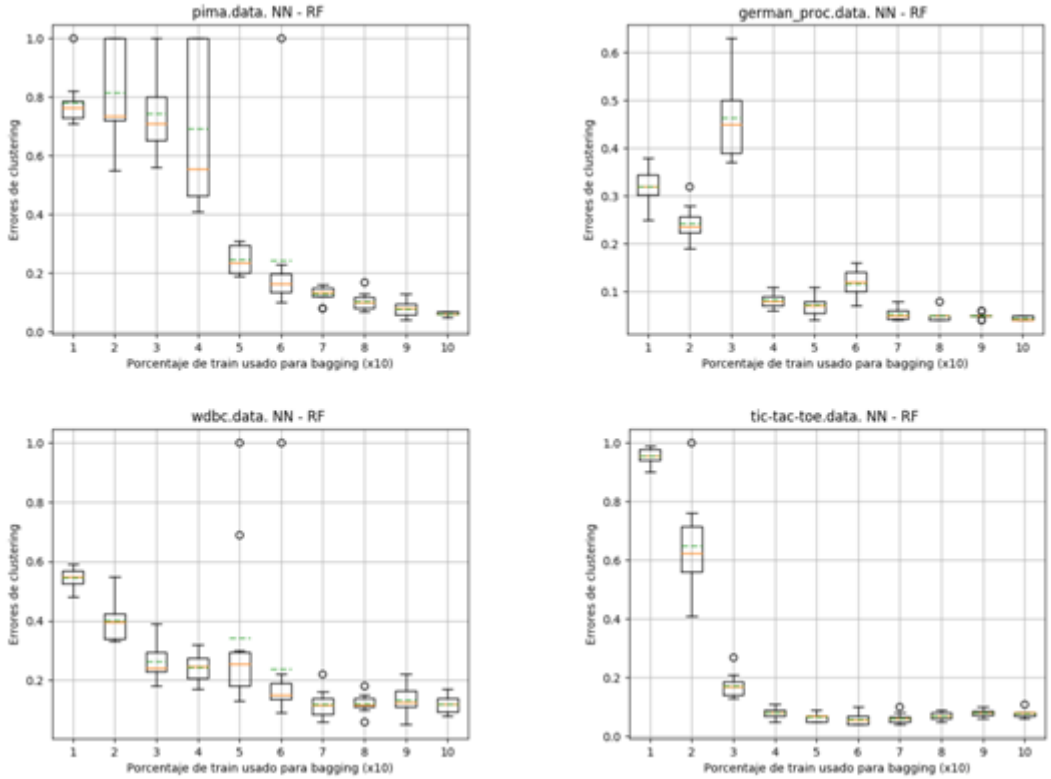

Fig 5-5. Grado de impureza de los clústeres frente a la tasa de remuestreo para un conjunto formado por perceptrones multicapa y árboles aleatorios

En este primer gráfico de la segunda parte de los experimentos se muestra la evolución del grado de impureza de los clústeres utilizamos los clasificadores MLP y RF sobre los mismos conjuntos de datos que anteriormente.

Analizando los propios resultados, nos encontramos con gráficas quizá menos semejantes entre ellas que en el caso anterior. Aunque en todas ellas la conclusión es parecida, cada una debe ser interpretada de manera separada.

Todas ellas muestran una evolución similar. Tal y como se anticipaba en la introducción de estos experimentos, el resultado es el esperado: el error de predicción es elevado para tasas de remuestreo pequeñas. A medida que aumenta dicha tasa, y, por consiguiente, el tamaño del conjunto de entrenamiento utilizado para entrenar los clasificadores individuales, este error tiende a disminuir hasta alcanzar un nivel asintótico, similar al del primer bloque de experimentos.

El uso del *boxplot* nos permite también analizar las desviaciones de los errores medios. Como es de esperar, los valores más dispersos se obtienen para tamaños pequeños del conjunto de entrenamiento. Este comportamiento confirma que en estos casos, la aleatoriedad debería tener un mayor impacto.

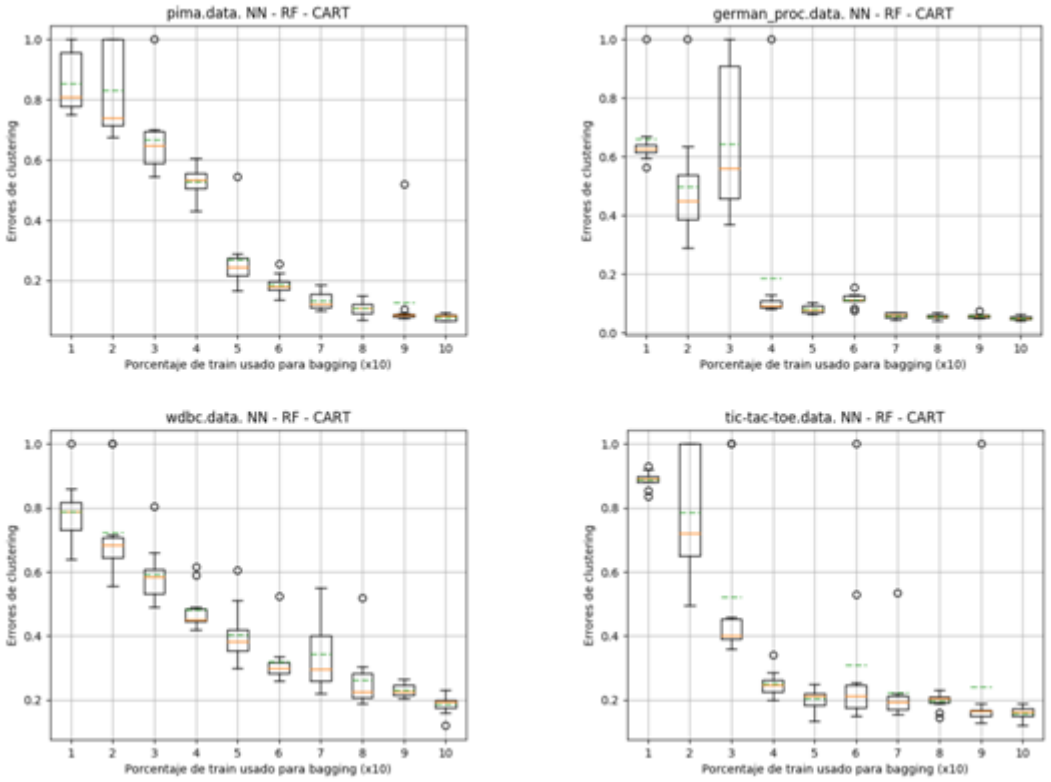

Fig 5-6. Grado de impureza de los clústeres frente a la tasa de remuestreo para un conjunto formado por perceptrones multicapa, árboles aleatorios y árboles de decisión

De igual manera que en los experimentos del apartado 4.2, el análisis en las figuras 5 y 6 se muestra la dependencia del grado de impureza con la tasa de remuestreo cuando se consideran conjuntos compuestos por tres tipos de clasificadores. En este caso se ha incluido un conjunto formado por árboles de decisión *CART*. Como ya fue comentado con anterioridad la elección del tercer clasificador es de menor importancia, ya que no será utilizado en el tercer bloque de experimentos, que son los más relevantes para los objetivos finales del trabajo. Únicamente son incluidos para aportar solidez a los resultados ya comentados.

Vemos en esta gráfica que las curvas presentan características similares a las anteriores, si bien son menos suaves. Este es un resultado esperado, pues la variabilidad del conjunto de partida es menor. No obstante, a pesar de este comportamiento, los clústeres finales también son bastante puros en este caso, lo que confirma que el algoritmo propuesto por (Bakker & Heskes, 2003) permite agrupar clasificadores del mismo tipo basándose únicamente en sus predicciones.

5.5 Uso de técnicas de clustering para poda de conjuntos

Los experimentos cuyos resultados han sido analizados en las secciones anteriores tienen como objetivo analizar el resultado de aplicar algoritmos de *clustering* a conjuntos de clasificadores, en los que cada clasificador es caracterizado por el patrón de predicciones en un conjunto de datos fijo.

Una de las posibles aplicaciones de este análisis es la poda de conjuntos. El objetivo de esta poda es identificar un subconjunto de los clasificadores del conjunto completo cuyas predicciones sean tan o más precisas que las del conjunto, De esta forma se puede mejorar tanto la eficacia y la eficiencia del sistema de predicción: podría obtenerse un menor error de clasificación empleando menos recursos computacionales. Para ello intentaremos seleccionar de entre los clústeres identificados por el algoritmo propuesto por (Bakker & Heskes, 2003) aquél cuyo error de predicción sea menor.

Identificaremos el mejor clúster c_k^* como aquél que minimice el error obtenido sobre $\boldsymbol{D}_{validation}$:

$$c_k^* = \underset{c_k \epsilon \{c_1,\ldots,c_K\}}{\arg\min} [E_{validation}(H_k)] \tag{16}$$

Sin embargo, nos encontramos con un problema: si seleccionamos el mejor clúster utilizando los errores que obtienen sus clasificadores sobre el conjunto de test, estamos utilizando información obtenida *a posteriori* para determinar algo que queremos conocer *a priori*. Por lo tanto, existe riesgo de sobre-aprendizaje: los resultados estarían sesgados y no podríamos garantizar que la capacidad de generalización del clúster seleccionado es superior a la de los otros clústeres.

Se pueden utilizar diferentes estrategias para evitar este sesgo. Una posibilidad es realizar una nueva partición del conjunto de datos etiquetados del que disponemos para realizar la inducción y obtener dos subconjuntos: un nuevo conjunto de entrenamiento y lo que se conoce como conjunto de validación. Generalmente, este segundo conjunto se utiliza para determinar los valores de hiperparámetros necesarios para realizar el entrenamiento. Nosotros adoptamos esta idea para adecuarla a nuestro problema. Tomaremos las predicciones que obtenga cada clasificador sobre el conjunto de validación (habiendo sido entrenado con el nuevo conjunto de entrenamiento que resulta de este segundo particionado) para realizar el *clustering* de los clasificadores. Una vez confeccionados los clústeres, mediremos el error de validación para cada uno de los clústeres y, finalmente, seleccionaremos el clúster que tenga el error mínimo.

Es importante señalar que la doble utilidad del conjunto de validación no supone riesgo de introducir sesgo alguno. Esto es debido a que ambas aplicaciones son totalmente independientes: el *clustering* se realiza sin atender a las predicciones reales, por lo que la medición del error de validación después del agrupamiento no intercede con dicho proceso.

El error general del subconjunto de clasificadores seleccionado puede ser estimado de manera no sesgada mediante el error en el conjunto de test. Este error en test es comparado con el error del conjunto de clasificadores heterogéneo completo y con el de los subconjuntos homogéneos de los que se compone.

El protocolo utilizado en los experimentos es el siguiente: para cada problema, el conjunto de datos etiquetados del que disponemos es dividido de manera aleatoria en un conjunto con 2/3 de los ejemplos para entrenamiento y validación y en un conjunto y un conjunto de test con los ejemplos restantes. El primer conjunto es particionado valiéndose de la proporción de 4/5 para entrenamiento y 1/5 para validación. A pesar del reducido tamaño del conjunto de la validación, este parece ser suficiente para formar los clústeres de forma coherente. Recordemos que únicamente lo utilizaremos para obtener las predicciones sobre las cuales agruparemos los clústeres más adelante. Para este tamaño, el desempeño del algoritmo de *clustering* no se ha visto afectado, y además los resultados encontrados en este apartado son igualmente satisfactorios. Además, lo conseguimos utilizando una parte mínima del conjunto de entrenamiento, y dejando así una buena cantidad de ejemplos que

permitirán entrenar sobradamente el clasificador. De todos modos, por esta razón, se ha creído conveniente utilizar conjuntos de datos de gran tamaño para asegurarnos que la proporción de validación no resulta demasiado reducida. Esta limitación no es un problema, pues el uso de conjuntos de datos pequeños tiende a desenlazar en resultados dispares que no suelen aportar información rigurosa.

Uno de los parámetros a determinar para el análisis de conglomerados es determinar el número óptimo de clústeres. En los extensos experimentos exploratorios realizados para dilucidar esta cuestión, se observa que este número óptimo de clústeres depende del problema de clasificación considerado. En algunos problemas se pueden conseguir buenos resultados solamente con 2 clústeres, lo cual, dadas las observaciones de los experimentos anteriores, corresponde de manera aproximada a utilizar solamente los conjuntos homogéneos. En otros este resultado se puede mejorar si se elige generar más clústeres que tipos de clasificadores. La elección de formar 5 clústeres permite alcanzar mejores resultados que cualquier otro valor de los considerados (3, 7 y 9) en el conjunto de problemas analizados. Para valores más pequeños, la calidad de las predicciones es similar pero los tamaños de los clústeres son mayores. En general, para más de 5 clústeres, la reducción del tamaño es excesiva y aumenta notablemente el error.

Para determinar si las conclusiones estudio son significativas, para cada problema de clasificación y configuración considerado se han repetido los experimentos con particiones independientes. Los resultados reflejados en la tabla 5-2 son la media y la desviación estándar sobre estas repeticiones. En esta tabla se consigna asimismo $T_{c_k^*}$, el tamaño del clúster seleccionado y el número de redes neuronales ($T_{c_k^*}^{MLP}$) y el número de árboles aleatorios ($T_{c_k^*}^{RF}$) de los que está compuesto.

Dataset	***Conjunto MLP (100)***	***Conjunto RF (100)***	***Conjunto MLP + RF (100+100)***	***Mejor clúster c_k^****	
	Error	**Error**	**Error**	$T_{c_k^*}$ $(T_{c_k^*}^{MLP} + T_{c_k^*}^{RF})$	**Error**
Diabetes (pima.data)	0.231 ± 0.023	0.241± 0.015	0.245 ± 0.019	45 (41 + 4)	**0.226** ± 0.019

German (german_proc.data)	0.269 ± 0.022	0.272 ± 0.013	**0.261** ± 0.011	44 (42 + 2)	0.267 ± 0.021
Breast cancer Wisconsin (wdbc.data)	**0.034** ± 0.009	0.057 ± 0.014	0.046 ± 0.011	50 (42 + 8)	0.042 ± 0.015
Tic-tac-toe (tic-tac-toe.data)	0.259 ± 0.028	0.113 ± 0.023	0.158 ± 0.032	45 (0 + 45)	**0.112** ± 0.023
Blood (blood_proc.data)	0.245 ± 0.014	0.266 ± 0.017	0.251 ± 0.009	48 (44 + 4)	**0.240** ± 0.017
Heart disease (heart.data)	0.435 ± 0.042	**0.409** ± 0.021	0.410 ± 0.020	42 (1 + 41)	0.430 ± 0.037
Liver (bupa.data)	0.411± 0.031	**0.348** ± 0.036	0.365 ± 0.031	47 (3 + 44)	0.380 ± 0.044
Chess (kr-vs-kp.data)	**0.020** ± 0.005	0.023 ± 0.006	0.023 ± 0.006	44 (38 + 6)	**0.020** ± 0.004
SPECT Heart (spect.data)	0.335 ± 0.035	0.333 ± 0.030	0.339 ± 0.028	40 (29 + 11)	**0.321** ± 0.033
Cars (cars.data)	0.106 ± 0.024	0.096 ± 0.012	0.123 ± 0.015	46 (0 + 46)	**0.095** ± 0.015

Tabla 5-3. Errores de clasificación de los distintos conjuntos y del mejor clúster

En estos experimentos se presentan los resultados para todos los conjuntos de datos analizados.

Esta experimentación se planteó en un primer momento como una exploración para determinar las posibles ganancias que podían obtenerse de la poda de conjuntos de clasificación con técnicas de *clustering*. Por lo tanto, los resultados no refutan o confirman ninguna hipótesis inicial.

Para algunos de los conjuntos de datos, el clúster que obtiene un menor error en las predicciones del conjunto de validación también consigue clasificar los ejemplos del conjunto de test mejor que el conjunto completo y que los conjuntos de clasificadores homogéneos. Este resultado es óptimo, pues conseguimos mejorar el error de test

empleando un conjunto de clasificadores de menor tamaño, lo cual se traduce en la utilización de menos recursos computacionales. Es decir, logramos mejorar la eficacia y la eficiencia del conjunto de forma simultánea.

Si bien este resultado no es el único que apreciamos, lo cierto es que alcanza una mayoría en el conjunto de resultados. Entonces, ¿podemos tomar la poda de conjuntos mediante *clustering* como un método efectivo de mejorar su eficacia en la clasificación? La cuestión es más complicada de lo que puede parecer, pues nos encontramos también con otro resultado que consideramos bueno (aunque no óptimo). Para otros conjuntos, el error medio obtenido por los clasificadores del mejor clúster es muy próximo (no exactamente igual, ya que los resultados han sido redondeados) al alcanzado por los demás conjuntos. Consideramos este producto como positivo porque, si bien no consigue mejorar la eficacia del resto de conjuntos, llega a igualarla utilizando menos clasificadores. Si tomamos el conjunto de buenos resultados como la unión de los óptimos y los descritos en este párrafo, nos encontramos con que estos conforman una mayoría absoluta de todos los casos analizados.

Como última consideración, es necesario comentar que existen casos en los que obtenemos errores inusualmente grandes. Si analizamos los conjuntos para los cuales esto se da, nos damos cuenta de que son precisamente los más pequeños en los que las estimaciones del error en el conjunto de validación son poco fiables.

A la hora de analizar la composición de c_k^*, llama la atención especialmente la pureza de los clústeres obtenidos. En prácticamente la totalidad de los casos, existe un tipo de clasificador que predomina ampliamente en c_k^*.

Para hablar de los resultados de la poda de forma concreta, es necesario referenciar (Suárez, et al., 2009). Observamos que el tamaño del mejor clúster parece colocarse ligeramente por encima del 20% del tamaño total del conjunto completo. Este es un resultado bastante coherente, pues si formamos 5 clústeres y estos se reparten los puntos de forma uniforme, es de suponer que la dimensión de cada clúster equivalga a dicho porcentaje del tamaño total. Este resultado nos sugiere que los dos clasificadores escogidos quizá obtengan mejores resultados por separado que de forma conjunta.

Como resumen de los resultados observados, el método propuesto permite reducir el tamaño de los conjuntos de clasificadores de manera significativa (aproximadamente un 20% del tamaño original) con pequeñas diferencias, tanto positivas como negativas, en las tasas de acierto. Estos resultados, aun siendo preliminares, permiten plantear la posibilidad de desarrollar futuras líneas de investigación interesantes, como se detallará en un capítulo posterior de esta memoria.

6. Conclusiones y trabajo futuro

La parte dedicada a las conclusiones ha sido dividida en dos apartados. El primero tratará las conclusiones propias del trabajo que se ha llevado a cabo y el segundo las concusiones de una reflexión personal sobre el proceso de aprendizaje realizado.

6.1 Conclusiones del proyecto.

Como conclusión general, los resultados obtenidos han sido satisfactorios. Observando los objetivos que planteábamos al principio de esta memoria y a la vista de los resultados obtenidos, podemos asegurar que hemos conseguido cumplir la mayoría de las metas establecidas inicialmente. En concreto, se ha conseguido reducir el tamaño de un conjunto heterogéneo de clasificadores manteniendo, en algún caso empeorando ligeramente y en otros mejorando la calidad de las predicciones respecto al conjunto original.

No sólo podemos considerar positivo el resultado por satisfacer (en mayor o menor medida) nuestros objetivos, sino también porque también permite plantear vías de continuación del trabajo que pueden resultar interesantes y fructíferas.

6.2 Conclusiones personales.

Además de las conclusiones que sacamos del trabajo realizado, me ha parecido apropiado incluir también una selección de mis conclusiones personales que considero más importantes.

A mi parecer, este proyecto no sólo ha resultado en un desenlace provechoso, sino que también ha supuesto una aportación de gran valor en mi formación. Nunca antes en la carrera había afrontado un trabajo de este tipo, y me ha supuesto una motivación especial.

Además de tratar sobre un tema que me interesa tanto como el aprendizaje automático, me ha permitido adquirir capacidades de trabajo y estudio que no había aprendido hasta

ahora en la carrera. No hablo solamente de conocimientos técnicos. También quiero referirme al trabajo de investigación en general. En mi opinión, la enseñanza de la mecánica de los estudios de investigación no tiene una presencia suficiente durante nuestra formación universitaria y ésta debería tomar un papel más protagonista. También es cierto que un solo trabajo de investigación no cualifica a nadie, pero a mí me ha estimulado a querer aprender más en profundidad acerca de ello y poco a poco adquirir las competencias necesarias para desenvolverme con soltura en este campo tan complicado de dominar.

Lo que más ha llamado mi atención de esta forma de trabajar es que, como se comentaba antes, ningún resultado es desechable. Todos ellos, por irrelevantes que parezcan, siempre desvelan información que siempre es útil. Ninguna idea debe ser despreciada, pues cada una puede utilizarse para añadir conocimiento o para guiar la investigación en alguna dirección. De esta manera, un proyecto de investigación puede durar hasta que el propio interesado así lo decida. Siempre surgen ideas nuevas a partir de los resultados que se van obteniendo.

Indudablemente, continuaré la investigación más allá de la entrega de mi proyecto y trataré de añadir más valor a este trabajo introduciendo más experimentos y con ellos más resultados valiosos. Las ideas que tengo en mente serán desarrolladas ampliamente en la siguiente sección.

6.3 Trabajo futuro.

Aunque el trabajo desarrollado haya terminado de forma productiva y satisfactoria, como ya adelantábamos, en este tipo de proyectos siempre surgen nuevas propuestas de investigación que pueden acabar trascendiendo aún más de lo ya conseguido.

A partir de los últimos experimentos se abren diversas opciones para continuar con el trabajo y llegar a alguna conclusión que le aporte un mayor interés. Además de nuevos experimentos, también se pueden incluir mejoras en los ya existentes. Primero hablaremos de dichas mejoras que pueden ser introducidas para después pasar a la exposición de las nuevas ideas.

Se puede valorar la posibilidad de experimentar con validación cruzada. La principal dificultad que encontramos es saber cómo se establece la correspondencia entre los clústeres obtenidos para cada partición. Suponiendo que los clústeres sean similares para todas ellas, se podría utilizar la composición de éstos para solventar dicha complicación.

En lo que respecta a los dos primeros experimentos, a mi parecer, poco queda por explorar más allá de lo ya realizado. Recordemos que estos experimentos tienen como objetivo analizar el comportamiento de los algoritmos de *clustering*. Con los resultados obtenidos las conclusiones son claras. Como posible profundización, podemos añadir más clasificadores, probar otros tipos de clasificadores o realizar experimentos con más conjuntos de datos. No obstante, de acuerdo con el análisis realizado, es de esperar que los resultados sean similares.

Sin embargo, del último experimento podemos idear nuevas propuestas de interés. Por ejemplo, sería deseable diseñar un procedimiento automático para determinar el número de clústeres óptimo. Es quizá la mayor carencia que tiene nuestro diseño.

En cuanto a propuestas para nuevos experimentos, entraremos a analizar lo que ocurre cuando el conjunto completo se agrupa en clústeres que formen "subconjuntos" y que obtengan clasificaciones haciendo voto por mayoría a nivel del propio clúster y después a nivel del conjunto completo de los clústeres. Es posible que de esta manera obtengamos un mejor resultado que utilizando simplemente el voto por mayoría en el conjunto completo. La idea sería identificar los clústeres que mejor rendimiento demostraran y hacer valer su voto por encima del de los demás de forma proporcional utilizando un sistema de voto ponderado. La ponderación de cada clúster vendría definida de forma proporcional al error que obtuviera sobre el conjunto de validación. Es complicado aportar más detalles acerca de la propuesta, pues de momento sólo podemos hacer suposiciones acerca del resultado que obtendremos.

Para concluir con esta sección, se abordará una cuestión de importancia sobre la que merece la pena detenerse. Desde el inicio, se acordó que el título del trabajo fuera "Métodos bayesianos para conjuntos de clasificación". Sin embargo, durante el desarrollo del mismo, se advirtió el gran interés que tenía el aplicar algoritmos de *clustering* como paso previo a un tratamiento bayesiano. Al observar los resultados y comprobar que llegábamos a

conclusiones interesantes, se decidió continuar por esa vía. Por este motivo, era necesario modificar el título del trabajo a "Métodos de *clustering* para conjuntos de clasificación". De esta forma, una más que probable vía de actuación futura será la de realizar inclusiones de carácter bayesiano en la metodología implementada para tratar de mejorar los resultados u obtener otros distintos.

Referencias

Ashton, K., 2009. That 'internet of things' thing. *RFID Journal*.

Bache, K. & Lichman, M., s.f. *UCI Machine Learning Repository.* [En línea] Available at: http://archive.ics.uci.edu/ml/index.php [Último acceso: Abril 2017].

Bakker, B. & Heskes, T., 2003. Clustering ensembles of neural network models. *Neural Networks,* Issue 2, pp. 261-269.

Bezdek, J. C., Ehrlich, R. & Full, W., 1984. FCM: The fuzzy C-Means clustering algorithm. *Computers & Geosciences,* Issue 10, pp. 191-203.

Breiman, L., 2001. Random Forests. *Machine Learning,* 45(1), pp. 5-32.

Breiman, L., Friedman, J., Stone, C. J. & Olshen, R., 1984. *Classification and Regression Trees.* Monterrey: Brooks/Cole Publishing.

Buhmann, J. M. & Kühnel, H., 1993. Vector quantization with complexity costs. *IEEE Transactions on Information Theory,* IV(39), p. 1133 – 1145.

Efron, B. & Tibshirani, R., 1993. *An introduction to the Bootstrap.* s.l.:Chapman & Hall/CRC.

Inza, I. y otros, 1999. Representing the behaviour of supervised classication learning algorithms by bayesian networks. *Pattern Recognition Letters,* Volumen 20, pp. 1201-1209.

Little, M. y otros, 2007. Exploiting non-linear recurrence and fractal scaling properties for voice disorder detection. *BioMedical Engineering Online,* pp. 6-23.

López Briega, R. E., 2015. [En línea] Available at: http://relopezbriega.github.io/blog/2015/02/01/programacion-funcional-con-python/ [Último acceso: Abril 2017].

MacQueen, J. B., 1967. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability,* Volumen I, pp. 281-297.

Mattern, F. & Floerkemeier, C., 2010. From the internet of computers to the internet of things. En: *From Active Data Management to Event-Based Systems.* Berlin: Springer Berlin Heidelberg, pp. 242-259.

Parekh, P. M., Katselis, D., Beck, C. L. & Salapaka, S. M., 2015. Deterministic Annealing for Clustering: Tutorial and Computational Aspects. *American Control Conference (ACC),* pp. 2906-2911.

Pedregosa, F. y otros, 2011. Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research,* Volumen 12, pp. 2825-2830.

Python Software Foundation, 2016. *Python documentation.* [En línea] Available at: https://docs.python.org/3/whatsnew/3.6.html [Último acceso: Abril 2017].

Rose, K., Gürewitz, E. & Fox, G., 1990. Statistical mechanics of phase transition in clustering. *Physical Review Letters,* Issue 65, pp. 945-948.

Rubin, D. B., 1991. EM and beyond. *Psychometrika,* II(56), pp. 241-254.

Suárez, A., Hernández-Lobato, D. & Martínez-Muñoz, G., 2009. An analysis of ensemble pruning techniques based on ordered aggregation. *IEEE Transactions on Pattern Analysis & Machine Intelligence,* Volumen 31, pp. 245-259.

Anexos

Anexo A: pseudocódigo

Function 1: Validation over D_{test} for experiments 1 and 2

Input:

D % Dataset

H % Classifiers array

T % Number of bags for classifier

S % Size of D_{train} used for training

F % Fraction of D used for partitioning

1 $D_{train}, D_{test} \leftarrow partition(D, F)$

2 **for** h **in** H **do**

3 **for** $t \leftarrow 1$ **to** T **do**

4 $bag_s \leftarrow bootstrap\_sample(D_{train}) \; \forall s = 1, \dots, S$

5 $train(h, bag)$

6 $p_t \leftarrow predict(h, D_{test})$

7 $c_t \leftarrow h$

Output: *P % Predictions =* $\{p_t\}_{t=1}^{T}$ *, C % Trained classifiers =* $\{c_t\}_{t=1}^{T}$

Function 2: Validation over $D_{validation}$ for experiment 3

Input:

D % Dataset

H % Classifiers array

T % Number of bags for classifier

S % Size of D_{train} used for training

F_1 % Fraction of D used for first partitioning

F_2% Fraction of D_{train} used for second partitioning

$D_{train}, D_{test} \leftarrow partition(D, F_1)$

$D^*_{train}, D_{validation} \leftarrow partition(D_{train}, F_2)$

for h **in** H **do**

  for $t \leftarrow 1$ **to** T **do**

    $bag_s \leftarrow bootstrap\_sample(D^*_{train}) \; \forall s = 1, \dots, S$

    train(h, bag)

    $p_t \leftarrow predict(h, D_{validation})$

    $c_t \leftarrow h$

Output: *P % Predictions =* $\{p_t\}_{t=1}^{T}$ *, C % Trained classifiers =* $\{c_t\}_{t=1}^{T}$

Function 3: Clustering with *fuzzy K-Means* algorithm

Input:

P % Predictions = $\{\boldsymbol{y}_t\}_{t=1}^{T}$

K % Number of clusters

$D(\boldsymbol{x}_1, \boldsymbol{x}_2)$ *% Distance function*

ε *% Termination criterion*

$\boldsymbol{m}_k \leftarrow random(P)\ \ \forall k = 1, \dots, K$

do

$(d_{tk} \leftarrow D(\boldsymbol{y}_t, \boldsymbol{m}_k)\ \ \forall t = 1, \dots, T)\ \ \forall k = 1, \dots, K$

$(u_{tk} \leftarrow \frac{\mathbf{1}}{\sum_{\boldsymbol{k}'} \frac{d_{tk}}{d_{tk'}}}\ \ \forall t = 1, \dots, T)\ \ \forall k = 1, \dots, K$

$\boldsymbol{m}'_k \leftarrow \frac{\sum_t \boldsymbol{y}_t u_{tk}^2}{\sum_i u_{tk}^2}\ \forall k = 1, \dots, K$

while $D(\boldsymbol{m}_k,\ \boldsymbol{m}'_k) > \varepsilon$

$c_k \leftarrow \{c_k, \boldsymbol{y}_t\} : k = \arg\max_{k'} u_{tk'}\ \ \forall t = 1, \dots, T$

Output: *C % Clusters* = $\{c_k\}_{k=1}^{K} : c_k = \{\boldsymbol{y}_t\} : \boldsymbol{y}_t\ \epsilon\ P\ \forall t$

Function 4: Clustering with (Bakker & Heskes, 2003) algorithm

Input:

P % Predictions = $\{\boldsymbol{y}_t\}_{t=1}^{T}$

K % Number of clusters

β *% Initial beta*

$\Delta\beta$ *% Beta increment*

$D(\boldsymbol{x}_1, \boldsymbol{x}_2)$ *% Distance function*

ε *% Termination criterion*

$\boldsymbol{m}_k \leftarrow random(P) \;\; \forall k = 1, \dots, K$

do

$(d_{tk} \leftarrow D(\boldsymbol{y}_t, \boldsymbol{m}_k) \; \forall t = 1, \dots, T) \;\; \forall k = 1, \dots, K$

$(p_{tk} \leftarrow \frac{e^{-\beta d_{tk}}}{\sum_{k'} e^{-\beta d_{tk'}}} \; \forall t = 1, \dots, T) \;\; \forall k = 1, \dots, K$

$\boldsymbol{m}'_k \leftarrow \frac{\sum_t p_{tk} \boldsymbol{y}_t}{\sum_t p_{tk}} \; \forall k = 1, \dots, K$

$\beta \leftarrow \beta + \Delta\beta$

while $D(\boldsymbol{m}_k, \; \boldsymbol{m}'_k) > \varepsilon$

$c_k \leftarrow \{c_k, \boldsymbol{y}_t\} : k = \arg\max_{k'} p_{tk'} \;\; \forall t = 1, \dots, T$

Output: *C % Clusters* = $\{c_k\}_{k=1}^{K} : c_k = \{\boldsymbol{y}_t\} : \boldsymbol{y}_t \in P \; \forall t$